

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

#3
0w
3/13 319902253 146-101
JC986 U.S. PTO
09/800507
03/08/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日

Date of Application:

2000年 3月13日

出願番号

Application Number:

特願2000-073926

出願人

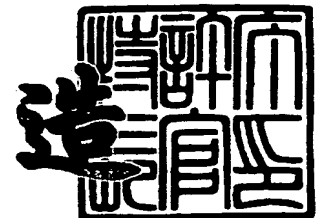
Applicant (s):

株式会社日立製作所

2000年10月 6日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2000-3081668

【書類名】 特許願

【整理番号】 H99022531A

【提出日】 平成12年 3月13日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 15/78

【発明者】

 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地
株式会社日立製作所中央研究所内

 【氏名】 荒川 文男

【発明者】

 【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地
株式会社日立製作所中央研究所内

 【氏名】 山田 哲也

【特許出願人】

 【識別番号】 000005108

 【氏名又は名称】 株式会社日立製作所

【代理人】

 【識別番号】 100075096

 【弁理士】

 【氏名又は名称】 作田 康夫

 【電話番号】 03-3212-1111

【手数料の表示】

 【予納台帳番号】 013088

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 データプロセッサ及びデータ処理システム

【特許請求の範囲】

【請求項 1】

SIMD型演算器を有するデータプロセッサであって、
前記データプロセッサは、前記SIMD型演算器にベクトルデータの処理を行わせるための命令を有していることを特徴とするデータプロセッサ。

【請求項 2】

請求項 1 において、
前記SIMD型演算器は、浮動小数点数の積和演算を行う複数の演算器を有していることを特徴とするデータプロセッサ。

【請求項 3】

命令セット内の命令を実行し、SIMD型演算器を有するデータプロセッサであって、
前記命令セットには、前記SIMD型演算器にベクトルデータの演算を行わせる命令が含まれていることを特徴とするデータプロセッサ。

【請求項 4】

請求項 3 において、
前記SIMD型演算器は、浮動小数点数の積和演算を行う複数の演算器を有していることを特徴とするデータプロセッサ。

【請求項 5】

命令セット内の命令を実行するデータプロセッサであって、
前記命令セットには、前記データプロセッサにベクトルデータの内積とスカラーデータとの和を演算させるための命令が含まれていることを特徴とするデータプロセッサ。

【請求項 6】

請求項 5 において、
前記データプロセッサは、4 元ベクトルと 4 元ベクトルとの内積を求め、前記内積とスカラーデータとの和を求める浮動小数点演算器を有していることを特徴

とするデータプロセッサ。

【請求項 7】

請求項 6 において、

前記浮動小数点演算器は、9 入力加算器を有していることを特徴とするデータプロセッサ。

【請求項 8】

命令セット内の命令を実行するデータプロセッサであって、

前記命令セットには、前記データプロセッサに、行列データとベクトルデータとの積を演算させるための命令が含まれていることを特徴とするデータプロセッサ。

【請求項 9】

請求項 8 において、

前記データプロセッサは、ベクトルデータとベクトルデータとの内積を求める浮動小数点演算器を複数個有していることを特徴とするデータプロセッサ。

【請求項 10】

請求項 8 において、

前記行列データは 4×4 の行列データであり、前記ベクトルデータは 4 元ベクトルであることを特徴とするデータプロセッサ。

【請求項 11】

請求項 9 において、

前記複数個の浮動小数点演算器のそれぞれは、前記内積とスカラーデータとの和とを求めることが可能な演算器であることを特徴とするデータプロセッサ。

【請求項 12】

請求項 11 において、

前記演算器は、9 入力加算器を有していることを特徴とするデータ処理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明はマイクロプロセッサ等のデータ処理を行うデータプロセッサ、更には

3次元グラフィック制御に好適なデータ処理システムに適用して、効率良く演算並列度を向上させる技術に関するものである。

【0002】

【従来の技術】

3次元グラフィックス処理を行うデータプロセッサとして、「MICROPROCESSOR REPORT, vol.13, no.5, April.19, 1999, pp.1, 6-11」では、1命令で4つの浮動小数点積和演算を実行するSIMD (Single Instruction Multiple Data) 型浮動小数点ユニットを2つ内蔵するプロセッサが示されている。ここで示されているプロセッサは、1つの浮動小数点積和演算で乗算と加算の2演算、4つで8演算、2ユニットでは計16演算を行うことが可能となっている。更に、上記の2ユニットの他に、通常の浮動小数点積和演算器が2つあるため、更に4演算を行うことが可能であり、合計20演算を1サイクルで行うことが可能となっている。

【0003】

また、上記以外のデータプロセッサとして、「IEEE Micro, vol.18, no.2, March/April 1998, pp.26-34」においては、浮動小数点の内積を求めるための命令を有し、1命令で4つの乗算と3つの加算を実行して、2つの4元ベクトルの内積を求めることが可能となるプロセッサが示されている。内積を求めるための命令の実行時には、1サイクルに7つの演算を行う。

【0004】

また、特開平10-124484においても、4つの乗算器に8つの浮動小数点数を与え、その乗算結果を4入力加算器で並列的に加算し、1回の並列的な乗算と加算とによって内積を求めることが可能なデータプロセッサが示されている。

【0005】

【発明が解決しようとする課題】

本発明者は、マルチメディア機器において、浮動小数点数を用いたグラフィック処理を従来以上に高速化することが可能なデータプロセッサ及び、データ処理システムについて検討を行った。

【0006】

マルチメディア機器向けデータプロセッサ、及びマルチメディア処理を行うデータ処理システムにおいて、重要かつ負荷の重い処理は、三次元グラフィクス処理と画像処理である。このうち、画像処理については規格化がなされているため、最も製造コストのかからない実現方法は専用ハードウェアを搭載することである。従来のプロセッサにおいても画像処理用の専用ハードウェアを搭載しているものは既に存在している。

【0007】

一方、三次元グラフィクス処理においては、座標計算等のジオメトリ処理と配色等のレンダリング処理とが必要となる。レンダリング処理については、汎用プロセッサ向きでなく、定型的な処理が一般的なため、高速処理が必要ならば専用ハードウェアを用いるのが一般的である。これに対して、座標計算等のジオメトリ処理については、自由度が高く浮動小数点データを扱うため、プロセッサの浮動小数点ユニットで処理するのが一般的である。このジオメトリ処理において最も頻繁に行われるのが4元ベクトル内積演算である。輝度計算は内積、座標変換は 4×4 行列と4元ベクトルの積、変換行列生成は 4×4 行列同士の積を求めることで処理される。それらの処理は、輝度計算が1回の4元ベクトル内積演算、座標変換が4回の4元ベクトル内積演算、変換行列生成については16回の4元ベクトル内積演算を行うことによって求めることができる。従来のプロセッサにおいて、4元ベクトル内積演算に特化して高速化を図り、効率的にジオメトリ処理の高速化を達成したものも存在している。

【0008】

しかし、三次元グラフィクス処理の高速化要求は極めて強く、動画像のリアリティを増すためには、更なる高速化が必要である。しかしながら、グラフィクス処理の基本データは4元ベクトルであるため、従来のプロセッサの方式では並列度をこれ以上上げることは困難である。FIR (Finite Impulse Response) フィルタ処理のように、多元ベクトル内積命令を定義すれば高速化されるアプリケーションも多いが、コンシューママルチメディア分野で最も高い浮動小数点演算性能が要求されるのは三次元グラフィクス処理である。従来知られているベクト

ル内積命令を有するプロセッサが効率的に並列度を向上したとしても、三次元グラフィクス処理の高速化に寄与しなければ意味がないものとなる。

【0009】

一方、SIMD方式では、原理的には並列度を上げることは容易である。しかし、SIMD方式は非効率な面も有しており、並列度を上げることによるコストは極めて増大する傾向にある。従来存在しているプロセッサにおいても既に大きな面積を割いているSIMD部分を更に何倍にもすることは現実的な解決方法とは言えない。例えば、従来技術で示した第1の文献のデータプロセッサでは、上述した性能を実現するために、実際に10個の浮動小数点積和演算器を内蔵しており、チップ面積は $0.25\mu\text{m}$ プロセスで製作したとしても、240平方ミリメートルと巨大な面積を必要とする。このうち、4つの浮動小数点積和演算を実行する並列SIMD型浮動小数点ユニットの面積をチップ写真から推定すると22平方ミリメートル程度となる。除算器を完全には4並列SIMD化していないことと、制御回路は必ずしも4倍にならないこと等により、通常の浮動小数点ユニットの面積の約3倍程度必要となる。

【0010】

また、従来技術で示した第2の文献で示されているデータプロセッサのチップ面積は、 $0.25\mu\text{m}$ プロセスで製作したとすれば56平方ミリメートル程度となる。このうち、浮動小数点ユニットの面積をチップ写真から推定すると、10平方ミリメートル程度となる。そして、内積命令用演算器を除いた面積は約7.5平方ミリメートル程度である。したがって、内積命令追加により浮動小数点ユニットの面積が約1.3倍になっていることとなる。

【0011】

本発明の目的は、演算並列度を効率的向上したデータプロセッサ及びデータ処理システムを提供することである。

【0012】

本発明の別の目的は、回路規模の増大を極力抑え、浮動小数点数の演算を高精度かつ高速に処理することが可能なデータプロセッサ及びデータ処理システムを提供することである。

【 0 0 1 3 】

【課題を解決するための手段】

本願において開示される発明のうち、代表的なものの概要を簡単に説明すれば下記の通りである。

【 0 0 1 4 】

すなわち、データプロセッサは、浮動小数点数の演算処理能力を向上させるため、S I M D型演算器を内部に構成し、前記S I M D型演算器にベクトルデータの処理を行わせるための単一の命令を有している。また、データプロセッサは、浮動小数点数の演算処理能力を向上させるため、S I M D型演算器を有し、前記S I M D型演算器にベクトルデータの演算を行わせるための命令が、命令セットの中に含まれているものである。更に、前記S I M D型演算器は、浮動小数点数の積和演算を行う複数の演算器を有しているものである。

【 0 0 1 5 】

また、データプロセッサの命令セットには、前記データプロセッサにベクトルデータの内積とスカラーデータとの和を演算させるための命令が含まれている。その命令により、データプロセッサは、例えば4元ベクトルと4元ベクトルとの内積と、前記内積とスカラーデータとの和を一つの命令により求めることを可能とするものである。前記演算を行うため、データプロセッサは、浮動小数点演算器を有しているものである。浮動小数点演算器は、処理能力を高めるためにS I M D型演算器とすることも出来る。

【 0 0 1 6 】

前記S I M D型演算器を構成している演算器或いは浮動小数点演算器は、ベクトルの内積とスカラーデータとの和を高速に演算するため、多入力加算器を有しているものである。3次元グラフィックス処理を高速に行うため、3次元グラフィックス処理において多用される4元ベクトル処理に特化したデータプロセッサにおいては、演算器は9入力加算器を有するものである。

【 0 0 1 7 】

また、データプロセッサの命令セットには、一つの命令によって前記データプロセッサに行列データとベクトルデータとの積を演算させるものである。この命

令により、データプロセッサは、 4×4 行列と 4 元ベクトルとの積を一つの命令で演算することが可能となる。データプロセッサは、前記命令を処理するため、ベクトルデータとベクトルデータとの内積を求める浮動小数点演算器を複数個有しているものである。3 次元グラフィックス処理において多用される 4×4 行列と 4 元ベクトルとを用いた演算処理を高速に行うことが可能となる。前記浮動小数点演算器のそれぞれは、内積とスカラーデータとの和とを求めることも可能なものである。また、前記演算器は、多入力加算器を有しているものである。

【 0 0 1 8 】

本発明の前記並びにその他の目的と新規な特徴等については、本願発明の明細書の記述及び添付の図面より明らかになるであろう。

【 0 0 1 9 】

【発明の実施の形態】

図 1 は本発明を適用したデータプロセッサ DP の構成図である。本実施例のデータプロセッサ DP は、整数を処理する能力を有する整数ユニットとしての中央処理装置 CPU、浮動小数点数の演算を行う SIMD 型浮動小数点ユニット (Single Instruction Multiple Data Floating-point Unit) FPU、命令キャッシュ ICA、データキャッシュ DCA、バスコントローラ BSC、複数の周辺モジュール PM、アドレス端子 AT、データ端子 DT 等を有している。中央処理装置 CPU と SIMD 型浮動小数点ユニット FPU とは、命令バス IB を介して命令キャッシュ ICA に接続されており、命令キャッシュより命令を取り込む。中央処理装置と SIMD 型浮動小数点ユニットとが取り込む命令を指定するためのアドレスは、命令アドレスバス IA を介して中央処理装置より与えられる。中央処理装置と SIMD 型浮動小数点ユニットとデータキャッシュ DCA とは、データバス DB に接続されている。データキャッシュには、データアドレスバス DA を介して中央処理装置よりデータアドレスが供給される。命令キャッシュ ICA とデータキャッシュ DCA とは、キャッシュコントローラを有しているが、図示は省略している。命令キャッシュ及びデータキャッシュは、データ信号やコントロール信号も伝達されるキャッシュバス CB を介してバスコントローラ BSC に接続されている。命令キャッシュにおいて、キャッシュミス等が生じた場合、外部

アクセスのための命令アドレスは外部アクセス命令アドレスバス E I A を介してバスコントローラに与えられる。また、データキャッシュにおいて、キャッシュミス等が生じた場合、外部アクセスのためのデータアドレスは外部アクセスデータアドレスバス E D A を介してバスコントローラに与えられる。バスコントローラは、キャッシュより送られた命令アドレス又はデータアドレスに従って、アドレス端子 A T 及びデータ端子 D T を介して接続される外部メモリなどをアクセスするために、外部バスサイクルを起動する。その後、バスコントローラは、外部メモリなどよりデータ端子に到着した命令或いはデータを、キャッシュバス C B を介して命令キャッシュ或いはデータキャッシュに供給する。また、バスコントローラには、特に制限されないが、タイマやシリアルコミュニケーションインタフェースコントローラ等の周辺回路 P M が周辺バス P B を介して接続されている。尚、本実施例のデータプロセッサは、特に制限されないが、単結晶シリコンのような一つの半導体基板上に形成されている。また、特に制限される訳ではないが、本実施例のデータプロセッサは、R I S C (Reduced Instruction Set Computer) アーキテクチャを有し、命令セットの中には浮動小数点命令を有する。浮動小数点命令は、メモリ効率を高めるために 1 6 ビット長であっても良く、命令数が多くなっても対応できるように 3 2 ビット長であってもよいが、それらに限定される訳ではない。

【 0 0 2 0 】

本実施例のデータプロセッサをマルチメディア機器、例えばビデオゲーム機器などへ組み込むことで、3次元グラフィックスを十分にサポートしたデータ処理システムを実現することが可能となる。

【 0 0 2 1 】

次に、図 1 に示したデータプロセッサ D P について詳細な説明を行う。本実施例のデータプロセッサは、面積を節約するために、S I M D 型浮動小数点ユニット F P U はメモリアドレッシング能力を持っていない。つまり、中央処理装置 C P U は S I M D 型浮動小数点ユニットに代わってメモリアドレッシング機能を有することとなる。そのため、中央処理装置は、S I M D 型浮動小数点ユニットのためにメモリからデータのフェッチを行うだけでなく、S I M D 型浮動小数点ユニッ

トのために浮動小数点命令を含む全ての命令をメモリからフェッチする。中央処理装置によってフェッチされた命令は、命令バスIBを介して中央処理装置とSIMD型浮動小数点ユニットとの双方に取り込まれてデコードされる。中央処理装置は、デコードした命令がCPU命令であった場合にはデコードした命令に従って整数処理を実行し、デコードした命令が浮動小数点命令であった場合にはSIMD型浮動小数点ユニットに代わってアドレッシング処理などを行う。SIMD型浮動小数点ユニットは、デコードした命令がCPU命令であった場合には命令を無視し、デコードした命令が浮動小数点命令であった場合にはデコードした命令に従って浮動小数点演算を行う。ここで、デコードした命令がロード又はストア命令である場合、中央処理装置はデータアドレスをデータキャッシュに出力してデータのロード又はストアを要求する。その要求に対してデータキャッシュDCAは、入力されたデータアドレスのデータをデータバスDBへロード又はデータバスからストアする。尚、ストアの場合も通常のコピーバックキャッシュの外部アクセスはリードであり、リードによってキャッシングされたキャッシュラインの一部にストア動作を行う。但し、キャッシングの際に有効かつ更新されたキャッシュラインがリプレースされた場合は、キャッシュラインを外部にコピーバックする。ロード又はストア命令の対象レジスタがSIMD型浮動小数点ユニットのレジスタであった場合、SIMD型浮動小数点ユニットはロード命令ならばデータバス上の値をSIMD型浮動小数点ユニットのレジスタに書き込み、ストア命令ならばデータバス上にSIMD型浮動小数点ユニット内のレジスタの値を出力する。尚、SIMD型浮動小数点ユニットによるロード及びストアに関しては図3を用いて詳述する。

【0022】

図2は図1のデータプロセッサDPのSIMD型浮動小数点ユニットFPUの構成図である。図中の命令デコーダIDECは、命令バスIBから供給される命令をデコードし、デコード結果に基づき制御信号CTRLを生成し、4つのベクトル浮動小数点ユニットV-FPUを制御する。各ベクトル浮動小数点ユニットV-FPUは128ビット幅を有するバスを介してデータバスDBに接続されている。それぞれの浮動小数点ユニットは、制御信号CTRLに従いロード/スト

ア及び浮動小数点演算の処理を実行する。データバスDBのバス幅については、128ビット、256ビット或いは512ビット等で構成することが可能である。512ビット幅にすると配線量は大きくなるが、各ベクトル浮動小数点ユニットが専用の128ビットバスを持てるため制御が簡単になる。但し、512ビット幅のデータバスを有効活用するにはデータキャッシュDCAを512ビット幅または128ビット幅の4バンクといった構成にする必要がある。データバスが128ビット幅の場合、データバスのビット分割または時分割が必要である。例えば、各ベクトル浮動小数点ユニットが32ビットアクセスをしてSIMD型浮動小数点ユニット全体で128ビットアクセスする場合はビット分割で対応する。一方、各ベクトル浮動小数点ユニットが128ビットアクセスをする場合は時分割が必要である。また、あるベクトル浮動小数点ユニットが128ビットアクセスを行い、他のベクトル浮動小数点ユニットについては動作行わせないためのNOP (non-operation) 命令を定義する方式もある。

【0023】

図3は図2で示したベクトル浮動小数点ユニットV-FPUの一つについて示した構成図である。浮動小数点演算命令実行時には32ビット4バンクレジスタファイルRGSTの2つの4バンクリードポートX、Yと1つの通常リードポートZからレジスタ値を演算ブロックEBLKへ転送し、演算ブロックでの演算処理を行う。演算ブロックでの演算処理後、演算用ライトポートVからレジスタファイルに演算結果を書き込む。ロード命令実行時には、転送ブロックTBLKはロード制御信号LDCによってデータバスDBにのせられたロードすべきデータを選択し、レジスタファイルの転送用4バンクライトポートUを介してレジスタファイルに書き込む。データの転送幅が128ビット未満の場合、ライトアライナWALNはデータを適宜アライメントしてUポートに送る。ストア命令実行の場合、転送ブロックTBLKはレジスタファイルRGSTの転送用4バンクリードポートWからストアデータを読み出し、バスドライブ信号BDCに基づいて読み出したデータをデータバスに転送する。この場合も、転送幅が128ビット未満の場合はリードアライナRALNによって適宜アライメントしてデータバスに送る。また、レジスタ間転送命令実行時には、転送ブロックは、リードポートWか

らレジスタ値を読み出し、ロード制御信号LDCによってリードポートWより読み出されたデータ選択し、ライトポートUからレジスタファイル書き込む。この場合も、転送幅が128ビット未満の場合、リードアライナまたはライトアライナで適宜アライメントを行いレジスタ間転送を行う。

【0024】

尚、レジスタの構成方法として、32ビット幅のレジスタファイルをレジスタ番号で4バンクに分割して定義する方法と、128ビット幅のレジスタをビット方向に4分割して定義する方法との2つが考えられる。前者の利点は32ビット幅で4元ベクトルを定義できることであり、後者の利点は1本のレジスタで4元ベクトルを定義できることである。しかし、後者の方法では、データがベクトルかスカラかによらず1本のレジスタに収まるため、レジスタリード・ライトやフォーディング処理が容易ではあるが、レジスタの使用効率や拡張性が悪くなる。例えば、128ビット幅のレジスタに対して内積命令を定義した場合、演算出力を格納するレジスタについても128ビットであるため、上位96ビットが無駄になってしまう。また、内積命令の定義に128ビット幅を使用してしまうと、内積命令のSIMD化が困難となる。例えば4並列にしようすると512ビット幅が必要となる。一方、前者の方式では32ビット幅で4元ベクトル内積命令が定義できるため、出力レジスタの無駄はなくSIMD化が容易である。以上より、レジスタファイルをレジスタ番号で4バンクに分割して4元ベクトルを定義し、その4元ベクトルに対して内積命令を定義し、レジスタファイルと内積命令とをSIMD化して並列度を上げることが可能であり、効率的に1命令あたりの並列度を上げることができる。本実施形態では前者の方法を採用することとする。

【0025】

図4は図3の32ビット4バンクレジスタファイルRGSTの構成図である。4つあるバンクのそれぞれのバンクBANK0、BANK1、BANK2、BANK3は、4リード、2ライト、32ビット、16本のレジスタファイルから構成される。つまり、レジスタ数は4バンク×16本で、合計64本となる。レジスタファイルは、6ビットで規定されるレジスタ番号の上位4ビットと、レジス

タライト制御装置であるWCUとWCVからのライトイネーブル信号、WEU [0]、WEU [1]、WEU [2] WEU [3]、WEV [0]、WEV [1]、WEV [2] WEV [3] によって制御されている。

【0026】

レジスタライト制御装置であるWCUは、ライトレジスタ番号WNUの下位2ビット42とライトサイズWSIZEとから、ライトするバンクを決定し、更にライト指示WRITUがアサートされたら、ライトすべきバンクのライトイネーブル信号WEUをアサートする。ライトサイズWSIZEが128ビットの場合は、ライトレジスタ番号WNUに関係なく全てのバンクのライトイネーブル信号をアサートする。ライトサイズが64ビットの場合は、ライトレジスタ番号WNUに応じてバンク0、1または2、3のライトイネーブル信号をアサートする。ライトサイズが32ビットの場合は、ライトレジスタ番号WNUに応じて4つのバンクのうち、1つのバンクを指定するためのライトイネーブル信号をアサートする。以上により、転送用4バンクライトポートUを介してレジスタにデータが書き込まれる。演算用ライトポートVを介してレジスタにデータを書き込む場合、演算用ライトポートは常に32ビットライトなので、レジスタライト制御装置WCVはライト指示WRITVがアサートされたら、ライトレジスタ番号WNVの下位2ビット42に応じて4バンクのうち、1つのバンクを指定するためのライトイネーブル信号WEVをアサートする。

【0027】

各バンクでは、リードレジスタ番号RNW、RNX、RNY、RNZの上位4ビット41によって、16本の内の1本の32ビットレジスタが指定される。本実施例では、バンクが4つあるため、合計4本のレジスタをリードして出力することが可能である。転送用4バンクリードポートWは、転送幅が128ビット未満の場合は転送ブロックTBLKがアライメントするので、各バンクの出力を直接出力する。4バンクリードポートX、Yポートは、ベクトル命令用の128ビット出力時は転送用4バンクリードポート同様、各バンクの出力を直接出力する。また、通常演算用の32ビット出力時にはレジスタ番号RNW、RNX、RNY、RNZの下位2ビット42を用い、セクタ432と433とにより、リー

ドすべきバンクを選択し $X[0]$ と $Y[0]$ とに出力する。通常リードポート Z は、通常の 32 ビットポートなので、常にレジスタ番号 RNW 、 RNX 、 RNY 、 RNZ の下位 2 ビット 42 を用い、セクタ 431 により、リードすべきバンクを選択して出力する。この結果、 $X[0]$ 、 $Y[0]$ 、 Z には 64 本のレジスタの任意の 3 本の値を載せることができる。

【0028】

なお、本実施例のレジスタファイル $RGST$ は、転送系の転送用 4 バンクライトポート U 、転送用 4 バンクリードポート W と、演算系の演算用ライトポート V 、4 バンクリードポート X 、 Y 、通常リードポート Z とにポートが分かれているため、スーパスカラや $VLIW$ (Very Long Instruction Word) アーキテクチャを適用することも可能である。

【0029】

図5は図3の演算ブロック $EBLK$ の構成図である。本実施例の演算ブロックは 2 つの浮動小数点 4 元ベクトルの内積と浮動小数点数との和を計算する。即ち、 $X[0] \times Y[0] + X[1] \times Y[1] + X[2] \times Y[2] + X[3] \times Y[3] + Z$ を計算する。符号処理部 SPP 、指数処理部 EP 、4 つの乗算部 ($MLP0$ 、 $MLP1$ 、 $MLP2$ 、 $MLP3$)、4 つのアライナ ($ALN0$ 、 $ALN1$ 、 $ALN2$ 、 $ALN3$)、 Z アライナ $ALNZ$ 、9 入力加算器 $ADDR$ 、正規化部 NOR から成る。正規化部 NOR は、正規化以外に、正数化や丸め処理も行う。

【0030】

入力される各 X 及び各 Y の浮動小数点数の符号部は符号処理部 SPP へ、指数部は指数処理部 EP へ、仮数部は乗算部 ($MLP0$ 、 $MLP1$ 、 $MLP2$ 、 $MLP3$) へ入力される。

【0031】

符号処理部 SPP は、 X 及び Y の符号が入力され、 $X \times Y$ 4 つの積の符号 $S[0]$ 、 $S[1]$ 、 $S[2]$ 、 $S[3]$ を EOR ゲートによって生成する。更に、その結果と Z の符号 SZ との EOR を取ることにより、4 つの積が Z と異符号かどうかをチェックし、チェック結果を $Inv[0]$ 、 $Inv[1]$ 、 $Inv[2]$

」、Inv [3] として各アライナ ALN0、ALN1、ALN2、ALN3 及び 9 入力加算器 ADDR に出力する。異符号の基準とした SZ は正数化前の符号となるので正規化部 NOR に送られる。

【0032】

指数処理部 EPP は、 $X \times Y$ の 4 つの積の指数部と Z の指数部との 5 つ項の中の最大指数 Emax を求めて正規化部に送出する。更に、Emax と各項の指数差を求め、Ediff [0]、Ediff [1]、Ediff [2]、Ediff [3]、Ediff Z として 5 つのアライナ ALN0、ALN1、ALN2、ALN3、ALNZ に出力する。詳細については図 6～10 で説明する。

【0033】

各乗算部 MLP0、MLP1、MLP2、MLP3 には、各 X 及び各 Y の仮数部が入力される。入力された X の仮数部及び Y の仮数部の積をキャリー保存形式で求め、MC [0] と MS [0]、MC [1] と MS [1]、MC [2] と MS [2]、MC [3] と MS [3] のペアとしてそれぞれ対応するアライナ ALN0、ALN1、ALN2、ALN3 に出力する。キャリー保存形式とは、キャリー伝播加算器で加算すると通常の 2 進数となる形式で、キャリー伝播なしで生成することができるため高速に生成できる。

【0034】

アライナ ALN0、ALN1、ALN2、ALN3 は、符号処理部 SPP からの異符号 Inv、及び指数処理部 EPP からの指数差 Ediff によって、乗算部 MLP0、MLP1、MLP2、MLP3 から出力されるキャリー保存形式の仮数部の積のアライメント及び論理反転を行う。各アライナの出力は、MCaln [0] と MSaln [0]、MCaln [1] と MSaln [1]、MCaln [2] と MSaln [2]、MCaln [3] と MSaln [3] のペアとして 9 入力加算器 ADDR に送られる。詳細は図 10 で説明する。Z アライナは、指数処理部からの指数差 Ediff Z によって、Z の仮数部 MZ のアライメントを行い、MZaln として 9 入力加算器に送る。詳細は図 11 で説明する。

【0035】

9 入力加算器は、MCaln [0]、MSaln [0]、MCaln [1]、M

S a l n [1]、M C a l n [2]、M S a l n [2]、M C a l n [3]、M S a l n [3]、及び M Z a l n を加算し、結果を M a c m として正規化部に送出する。異符号の項の加算のために、4つのアライナ A L N 0、A L N 1、A L N 2、A L N 3 で4つのキャリー保存形式ペアの論理反転が行われるが、符号反転のためには更に+1する必要がある。+1は9入力加算器へのキャリーインで行うため、I n v [0]、I n v [1]、I n v [2]、I n v [3] を入力してキャリーイン数を制御している。詳細は図12、14を用いて説明する。

【0036】

正規化部 N O R は、符号生成部からの S Z、指数処理部からの E m a x、9入力加算器からの M a c m を受け取り、通常の浮動小数点演算器と同様に正規化、正数化及び丸めを行い、最終演算結果 V を生成してレジスタファイル R G S T に送る。

【0037】

図6は、指数処理部 E P P についての説明である。この例では、高速化のために指数差を4入力加算器で直接求めている。一般的には、4つの積の指数部を求め、これと Z の指数部の中から最大指数部を求め、さらに各指数部と最大指数部の差をとって指数差を求める手法が考えられるが、低速である上に、4つの加算器、4～10個の大小比較器（比較の並列度によって異なる）、5つの減算器が必要となり、論理規模が削減される訳ではない。本実施形態の指数処理部は、図6に示したように、指数差生成部 E D G と出力選択部 E O S とから成る。

【0038】

図7は、指数差生成部 E D G の構成である。5項から選び得る全ての2項の指数差を10個の4入力加算器 F A D R で求めている。例えば、 $X[0] \times Y[0]$ と $X[1] \times Y[1]$ の指数差は、 $EX[0] + EY[0] - EX[1] - EY[1]$ である。各指数部は、規格に基づくバイアス“127”がかかっている。しかし、上式では4つのバイアスが打ち消し合うので上式で求めた指数差のバイアスは“0”である。一方、 $X[3] \times Y[3]$ と Z の指数差は、 $EX[3] + EY[3] - EZ - 127$ である。バイアスが打ち消されるように-127を行う。

【0039】

図8は図7の4入力加算器FADRの詳細を示した図である。4つの入力IPT0、IPT1、IPT2、IPT3とが入力される8ビット4入力キャリー保存加算器FADRSと、9ビットキャリー伝播加算器FADRPから成る。図7では4項のうち2項を論理反転しただけなので、この2項の符号反転のために、2ビットのキャリーインがある。指数差の範囲を $-510 \sim 510$ で表現するには符号付き2進数で10ビットが必要なため、入力を10ビットまで符号拡張して10ビット幅で計算するのが単純である。しかし、実際には上位ビットは冗長であるため図8に示した構成とする。本実施形態の構成では、ビットキャリー伝播加算器FADRPからのキャリーアウトは4入力加算結果が正の場合に1となる。したがって、キャリーアウトは符号反転しなかった指数の方が大きい2つの指数が等しいことを表わしている。このキャリーアウトをGE信号として出力し、指数部の大小判定に使用する。尚、8ビット4入力キャリー保存加算器は図15のような1ビット4入力キャリー保存加算器を8個並べることにより構成される。

【0040】

図9は図6の出力選択部EOSの詳細を示した図である。まず、620において、指数差生成部EDGから出力される10本のGE信号（GE01、GE02、GE03、GE12、GE13、GE0Z、GE1Z、GE2Z、GE3Z）に基づき、選択制御信号sel0、sel1、sel2、sel3、selzを生成する。sel0、sel1、sel2、sel3、selzのそれぞれは、 $X[0] \times Y[0]$ 、 $X[1] \times Y[1]$ 、 $X[2] \times Y[2]$ 、 $X[3] \times Y[3]$ 、Zが最大指数を持つことを表わしている。但し、指数部が等しい場合は、この順に優先度があり、選択制御信号はホットワンであることが保証されている。例えば、全ての指数部が等しい場合はsel0のみがアサートされる。次に、630において、上記で生成された選択制御信号を基に最大指数Emaxを生成する。4入力加算器FADRで指数差を直接生成したため、最大指数は630において改めて生成することとなる。図のように、631及び632において、選択制御信号で指数部を選択し、8ビットキャリー伝播加算器633で加算する。尚、EZを選択した場合は、他の場合とバイアスを合わせるために $EZ + 12$

7を計算する。次に、640～644において、各項の指数差 $E_{diff}[0]$ 、 $E_{diff}[1]$ 、 $E_{diff}[2]$ 、 $E_{diff}[3]$ 、 $E_{diff}Z$ を求める。 E_{max} を求める前に指数差を計算しているため、指数差が「 E_{max} －各項の指数」となっていないため、符号反転しなければならない場合が存在する。このため、指数差生成部EDGから出力される指数差の一部については、セレクタ入力前に論理反転させている。更に、自分自身が最大指数であった場合は指数差は“0”なので、“0”を入力している。また、符号反転を完結させるには論理反転後に+1する必要がある。そこで、+1が必要であることを示す信号 $E_{diffp1}[0]$ 、 $E_{diffp1}[1]$ 、 $E_{diffp1}[2]$ 、 $E_{diffp1}[3]$ 、 $E_{diffp1}Z$ を、650によって求める。尚、本実施例では、指数差に対する+1は行わず、指数差を使用するアライナにおいて、+1が必要な場合には+1の代りに1ビットシフトしている。

【0041】

図10において、アライナALN0、ALN1、ALN2、ALN3について説明する。本実施例のアライナは、キャリー保存形式の4つの積のアライナである。キャリー側MC[n]（nは0から3までの数）用とサム側MS[n]用とにアライナが2つあり、同一の信号で制御される。まず、シフタSFT1、SFT2により、指数差 $E_{diff}[n]$ 分だけ右シフトを行う。次に $E_{diffp1}[n]$ が1の場合、1ビットシフタSFT1'、SFT2'において、更に1ビットシフトする。そして、662及び672においては、Inv[n]が1の場合、論理反転を行う。以上により、MC[n]に対応してはMCaln[n]を出力し、MS[n]に対応してはMSaln[n]を出力する。

【0042】

図11はZアライナALNZである。まず、シフタSFTZに入力されるMZを指数差 $E_{diff}Z$ だけ右シフトする。次に、1ビットシフタSFTZ'において $E_{diffp1}Z$ が1の場合、更に1ビットシフトを行い、結果をMZalnとして出力する。尚、異符号判定をZを基準に行っているため、Zの符号反転は不要である。

【 0 0 4 3 】

図 1 2 においては、図 5 で示した 9 入力加算器 A D D R について詳細について説明する。本実施例では、9 入力加算ではあるがキャリー保存形式によって 5 入力 が 9 入力になっているため、桁数増加は最大 3 ビットである。まず、アライナ A L N 0、A L N 1、A L N 2、A L N 3、A L N Z から出力される、9 つの入力のそれぞれを符号拡張部 S E において 3 ビット符号拡張する。符号拡張部の出力は、9 入力キャリー保存加算器アレイ C S A に入力される。次に、符号反転項数に応じたキャリーインを行う。キャリー保存形式の積の 1 ペアを反転するのに 2 ビットのキャリーインを行うため、4 つの積に対して最大 8 ビットのキャリーインが必要である。符号反転は I n v [0]、I n v [1]、I n v [2]、I n v [3] で制御しているので、図のようにこの 4 信号のそれぞれに 2 ビットのキャリーを対応させる。そして、9 入力キャリー保存加算器アレイ C S A に 6 ビット、キャリー伝播加算器 C P A に 2 ビットのキャリーインを行う。キャリー伝播加算器は、上記の 2 ビットのキャリーと、9 入力キャリー保存加算器アレイからのキャリー出力 C O U T と、サム出力 S O U T とから、正規化前仮数部 M a c m を生成する。

【 0 0 4 4 】

図 1 3 は、図 1 2 で示した 9 入力キャリー保存加算器アレイ C S A の詳細図である。図中左側から 3 ビット符号拡張部の出力が入力される。まず、1 段目を 3 入力キャリー保存加算器 7 3 0、7 3 1、7 3 2 とし、2 段目も 3 入力キャリー保存加算器 7 3 3、7 3 4 とし、3 段目を 4 入力キャリー保存加算器 7 3 5 とする。この構成により、項数が 9 から 6、4、2 と減少し、最終的にキャリー出力 C O U T 及びサム出力 S O U T を得る。また、本実施例では、キャリーインも、C I 0、C I 1、C I 2、C I 3、C I 4、C I 5 の 6 ビットまで可能である。尚、3 入力キャリー保存加算器は、例えば、図 1 4 のような 1 ビット 3 入力キャリー保存加算器をビット幅だけ並べて構成する。また、4 入力キャリー保存加算器は、例えば図 1 5 のような 1 ビット 4 入力キャリー保存加算器で構成する。

【 0 0 4 5 】

図 1 6 には、本発明の一つの実施形態であるデータ処理システムのブロックダ

イヤグラムを示している。同図において、上述したデータプロセッサDPはバスBUSに接続されている。ここで示したバスは、アドレスが転送されるバスとデータが転送されるバスとを含むものとする。また、バス幅等について制限されるものではない。更に、命令とデータとが同一のバスを転送される構成であっても、命令とデータとが別のバスを介して転送される構成であっても、同図で示されるバスに含まれるものである。それらバス構成は、データ処理システムにおける処理速度、面積効率、或いは接続されるデバイスの構成等によって様々に変更可能である。

【 0 0 4 6 】

上記バスBUSには、データプロセッサの作業領域やデータの一時記憶領域として使用されるSRAMや、データプロセッサのOS (Operating System) 等が記憶されるROMが接続されている。また、バスには、制御回路DMCを介してDRAMが接続されている。制御回路DMCは、DRAMに対してアドレスマルチプレクス制御やリフレッシュ制御等を行うものとするが、DRAM内部やデータプロセッサ内部に分散させる構成であってもよい。更に、バスには周辺装置制御部PDCと表示コントローラDCとが接続されている。周辺装置制御部には、光ディスク等の外部記憶装置ESDやキーボードKBD等が接続されている。また、表示コントローラには、表示装置DPが接続されている。

【 0 0 4 7 】

上記データプロセッサは、浮動小数点演算を実行するため命令を備えており、命令の転送のための浮動小数点演算のためのレジスタを有しているため、3次元グラフィック処理に多用される浮動小数点数の演算を高速で実行することが可能である。従って、マルチメディア機器であるゲーム機や携帯情報端末等として利用される本実施形態のデータ処理システムは、全体のコストを削減し、3次元グラフィック処理を高精度かつ高速に処理することが可能となる。

【 0 0 4 8 】

また、図16のデータ処理システムにおいて、バスBUSにレンダリングコプロセッサを追加することも可能である。3次元グラフィックス処理は、ジオメトリ処理とレンダリング処理とで構成される。内積演算やベクトル変換演算を多用

するジオメトリについては、本実施形態のデータプロセッサDPによって処理させ、レンダリング処理については、レンダリングコプロセッサに処理させる。このことにより、レンダリング処理をデータプロセッサ内の中央処理装置に処理させるデータ処理システムよりも、3次元グラフィックス処理を高速に処理することが可能なデータ処理システムを提供することが可能となる。

【0049】

以上、本発明者によってなされた発明を実施形態に基づいて具体的に説明したが、本発明は記載している実施形態に限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能であることは言うまでもない。

【0050】

例えば、データプロセッサは、メモリマネジメントユニットなど本実施形態で示したブロック以外のブロックを含むことも可能であり、本実施形態で示したブロックの配置等を変更することも可能である。また、データプロセッサは、スーパースカラアーキテクチャを採用することも可能である。スーパースカラアーキテクチャを採用することで、1本のパイプでは演算処理の命令を実行し、別のパイプではデータのメモリからのロードやメモリへのストアを行うことが可能となり、高速処理が可能となる。

【0051】

また、本実施形態においては、データプロセッサに形成されたFPUについて記載してきた。しかし、本発明の思想はFPUに限定されるものではなく、整数演算ユニットへの適用も可能である。整数演算ユニットへの適用の際、図5の乗算アレイ及び9入力加算器を用いれば整数型の4元ベクトルの内積とスカラの和との演算器を実現することが可能となる。整数型演算器の場合、純粋な16並列SIMD方式に比べて、演算器の論理規模の大幅削減はできないが、同等の演算並列度を1/4の4並列SIMDで達成することが可能となる。また、レジスタの幅も1/4で良いのでレジスタの論理規模を大幅に削減することが可能となる。

【0052】

以上、本願で開示した本発明の構成及び動作によって、上述した効果及び下記

に示す効果を得ることができる。

【0053】

本発明のデータプロセッサ及びデータ処理システムは、その命令セットの中に、図1のデータプロセッサに搭載されたSIMD-FPUにおいて4元ベクトルの内積とスカラとの和を浮動小数点形式で4つ処理させるための命令を持たせることが可能となる。上記構成をとり、上記命令を持たせることで、1サイクルで1命令を実行する場合、1サイクルで28の浮動小数点演算の実行が可能である。

【0054】

更に、SIMD-FPUのそれぞれの演算器を、4元ベクトルの内積とスカラーデータとの加算を行うことを可能とする構成とし、更に、データプロセッサの命令セットに内積と加算とを行わせる命令をもたせることで、多元ベクトルデータに対応するためことも可能となる。

【0055】

上記に示した、演算器の内積と加算の処理、と演算器のSIMD化とを組み合わせ、対応する命令を命令セットに加えることで、1サイクルに上記1命令を実行した場合、1サイクルに32の浮動小数点演算を実行することが可能となる。

【0056】

上記データプロセッサ、及び上記データプロセッサを用いて構成したデータ処理装置は、従来のデータプロセッサ及びデータ処理システムに対し、大幅に演算並列度を上げることが可能となり、処理速度を向上させることが可能となる。上記データプロセッサは、1サイクルあたり32FLOPSの処理能力を有することとなる。

【0057】

また、図5に例示している如く、本発明の演算ブロックでは、仮数部の処理及び最大指数項と各項の指数差の計算に、一度だけキャリー伝播加算を行うように構成している。そのため、演算レイテンシの短縮が容易で高周波数動作に適している。

【 0 0 5 8 】

【発明の効果】

本発明によって4元ベクトル内積命令を4並列のSIMD化することにより、1命令で28演算の実行が可能である。更に、4元ベクトルの内積とスカラの和の演算を定義することにより4元を超える多元ベクトルデータに対応する事が可能となり、1命令で32演算の実行が可能となる。故に、浮動小数点数の演算を高速で処理可能なデータプロセッサを提供することが可能となり、更に、マルチメディア処理特に3次元のグラフィック処理を高速に処理可能なデータ処理システムの提供が可能となる。

【図面の簡単な説明】

【図1】

本発明を適用したプロセッサの構成図。

【図2】

本発明を適用したプロセッサのSIMD-FPUの構成図。

【図3】

SIMD-FPUのベクトルFPUの構成図。

【図4】

ベクトルFPUの32ビット4バンクレジスタファイルの構成図。

【図5】

ベクトルFPUの演算ブロックの構成図。

【図6】

演算ブロックの指数演算部。

【図7】

指数演算部の指数差生成部。

【図8】

指数差生成部の4入力加算器。

【図9】

指数演算部の出力選択部。

【図 1 0】

演算ブロックのアライナ。

【図 1 1】

演算ブロックのZアライナ。

【図 1 2】

演算ブロックの9入力加算器。

【図 1 3】

9入力加算器の9入力キャリー保存加算器アレイ。

【図 1 4】

1ビット3入力加算器の例。

【図 1 5】

1ビット4入力加算器の例。

【図 1 6】

本発明のデータプロセッサを使用したデータ処理システム。

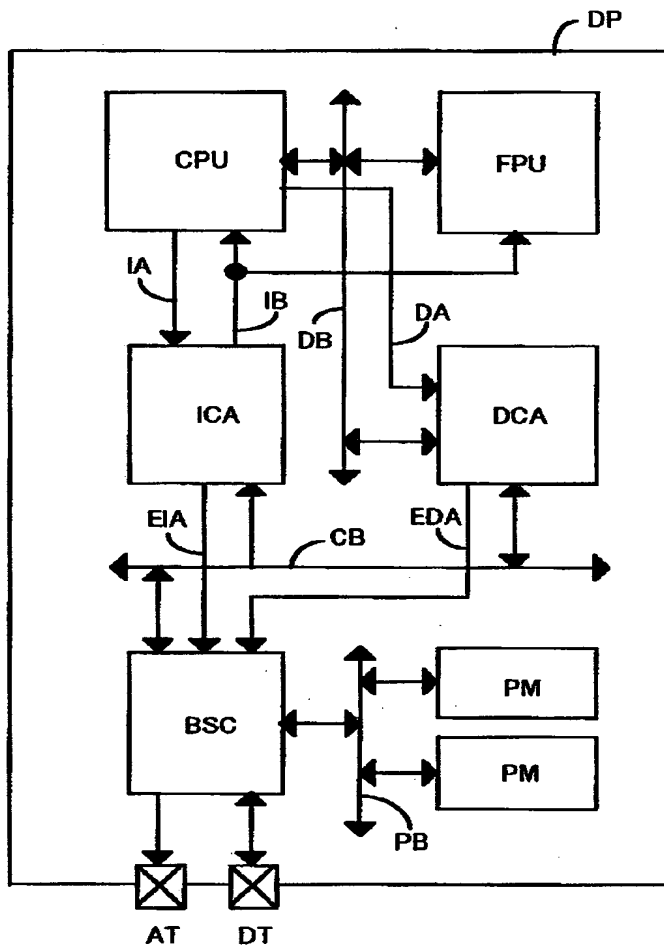
【符号の説明】

DP：データプロセッサ、CPU：中央処理装置、FPU：浮動小数点ユニット、ICA：命令キャッシュ、DCA：データキャッシュ、BSC：バスコントローラ、PM：周辺モジュール、IA：命令アドレスバス、IB：命令バス、DA：データアドレスバス、DB：データバス、CB：キャッシュバス、EIA：外部アクセス命令アドレスバス、EDA：外部アドレスデータアドレスバス、PB：周辺バス、AT：アドレス端子、DT：データ端子、V-FPU：ベクトル浮動小数点ユニット、IDEC：命令デコーダ、CTRL：制御信号、EBLK：演算ブロック、TBLK：転送ブロック、RGST：レジスタファイル、BDC：バスドライブ信号、WALN：ライトアライナ、LDC：ロード制御信号、RALN：リードアライナ、W(0)：リードポート、V：演算用ライトポート、X(0)：リードポート、Y(0)：リードポート、U：ライトポート、Z：リードポート、WCU：レジスタライト制御装置、ADDR：9入力加算器、SPP：符号処理部、EPP：指数処理部、NOR：正規化部。

【書類名】 図面

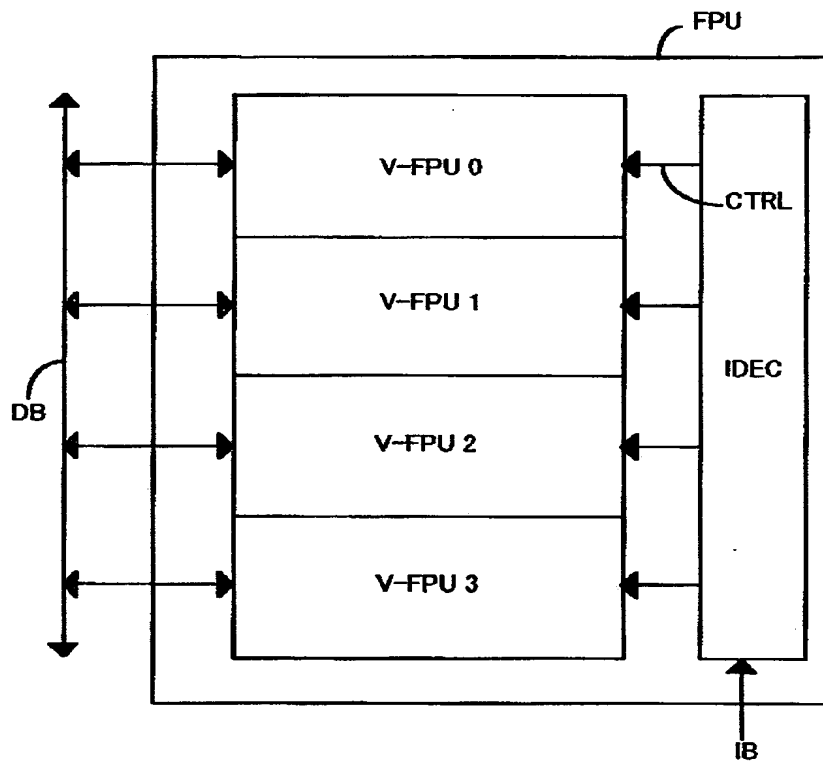
【図 1】

図 1



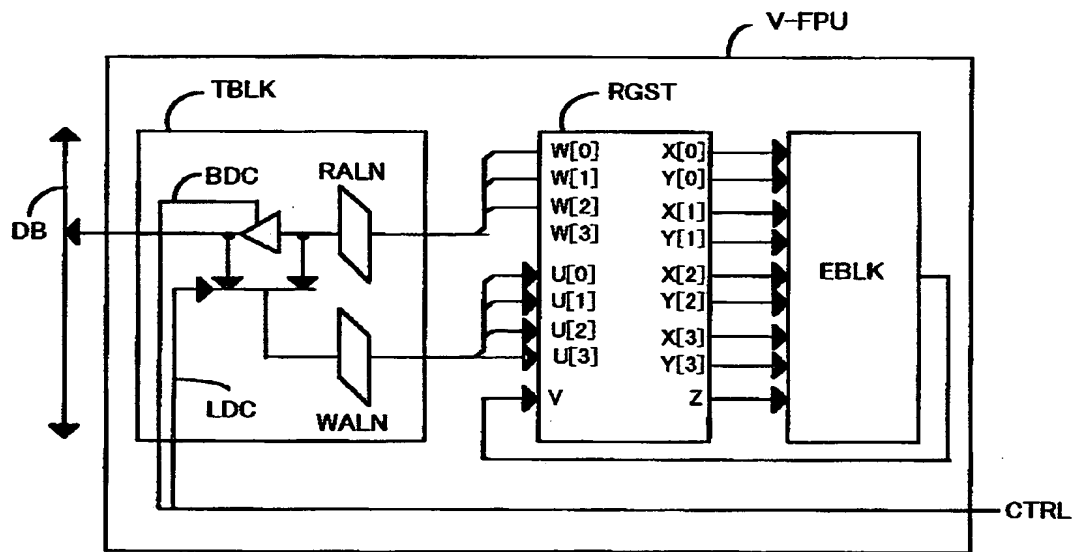
【図 2】

図 2



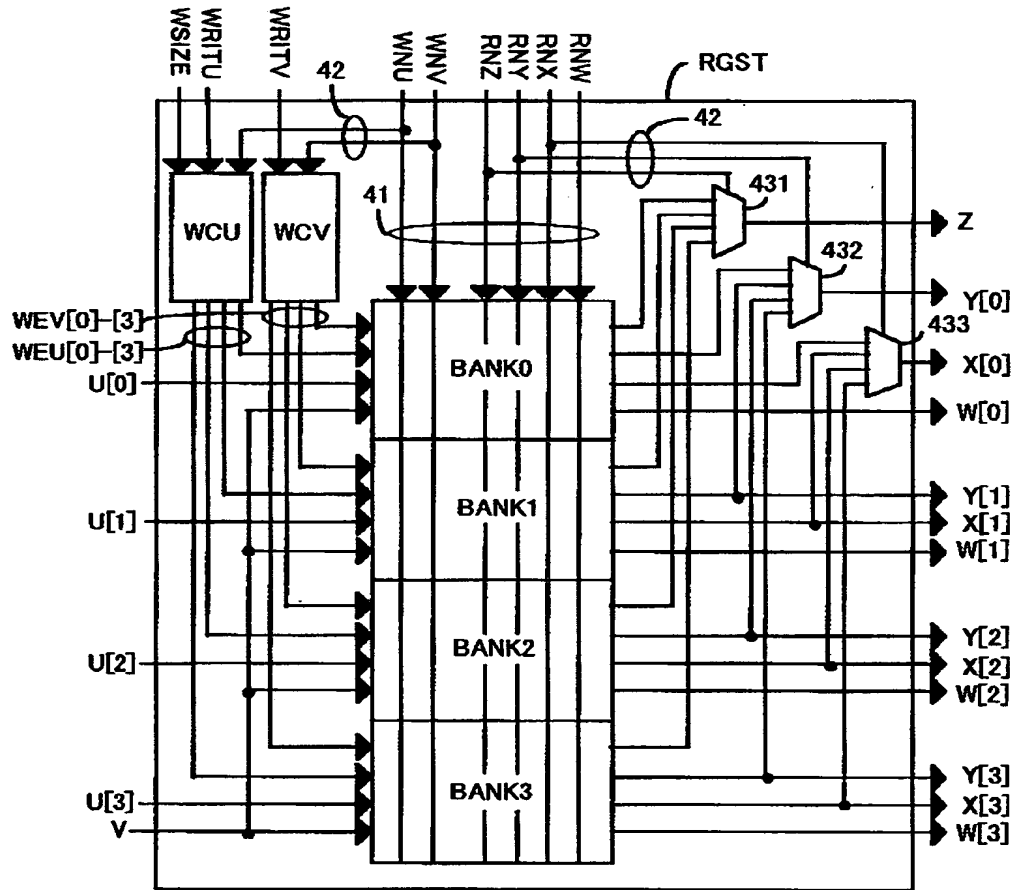
【図 3】

図 3



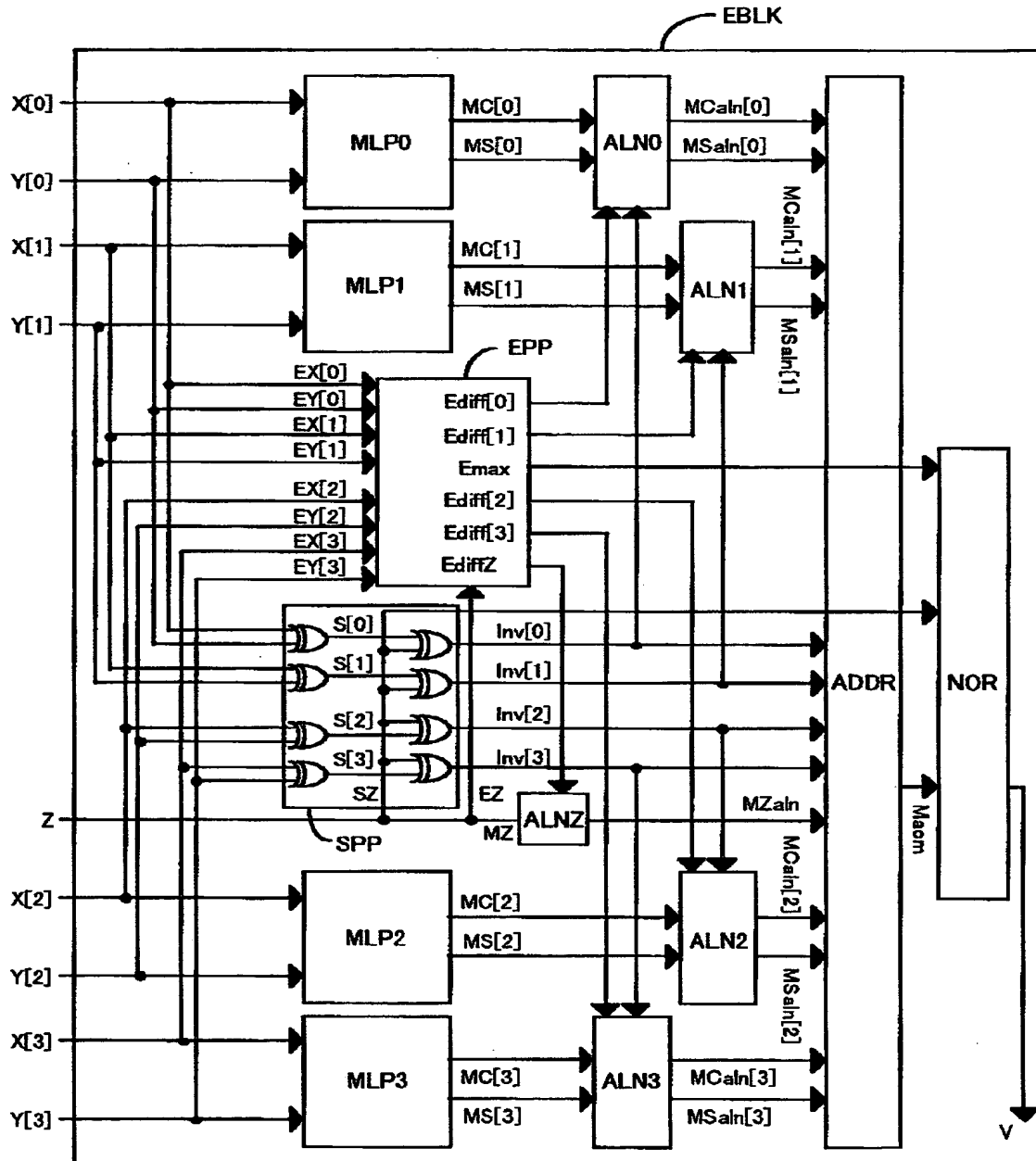
【図 4】

図 4



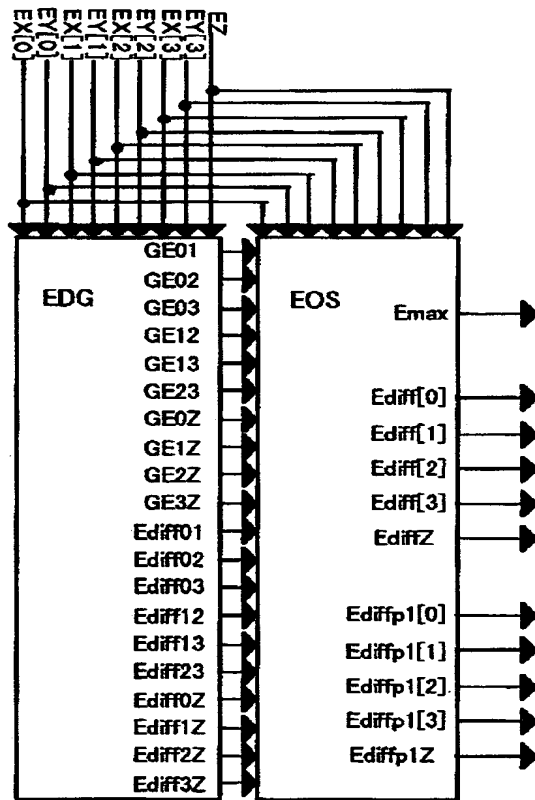
【図 5】

図5



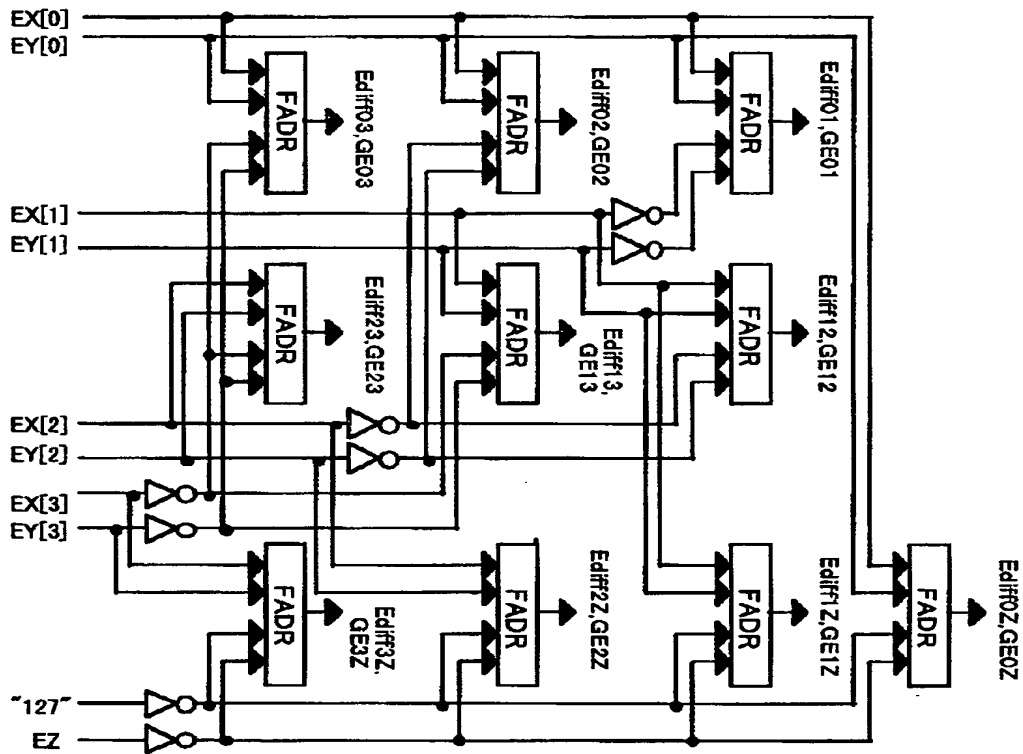
【図 6】

図 6



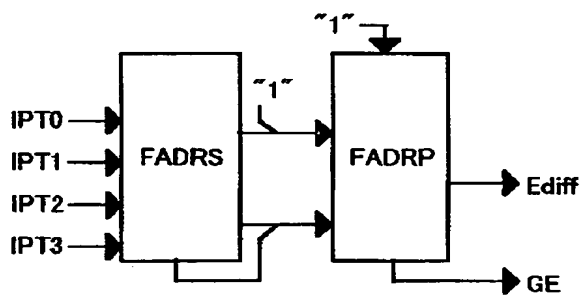
【図 7】

図 7



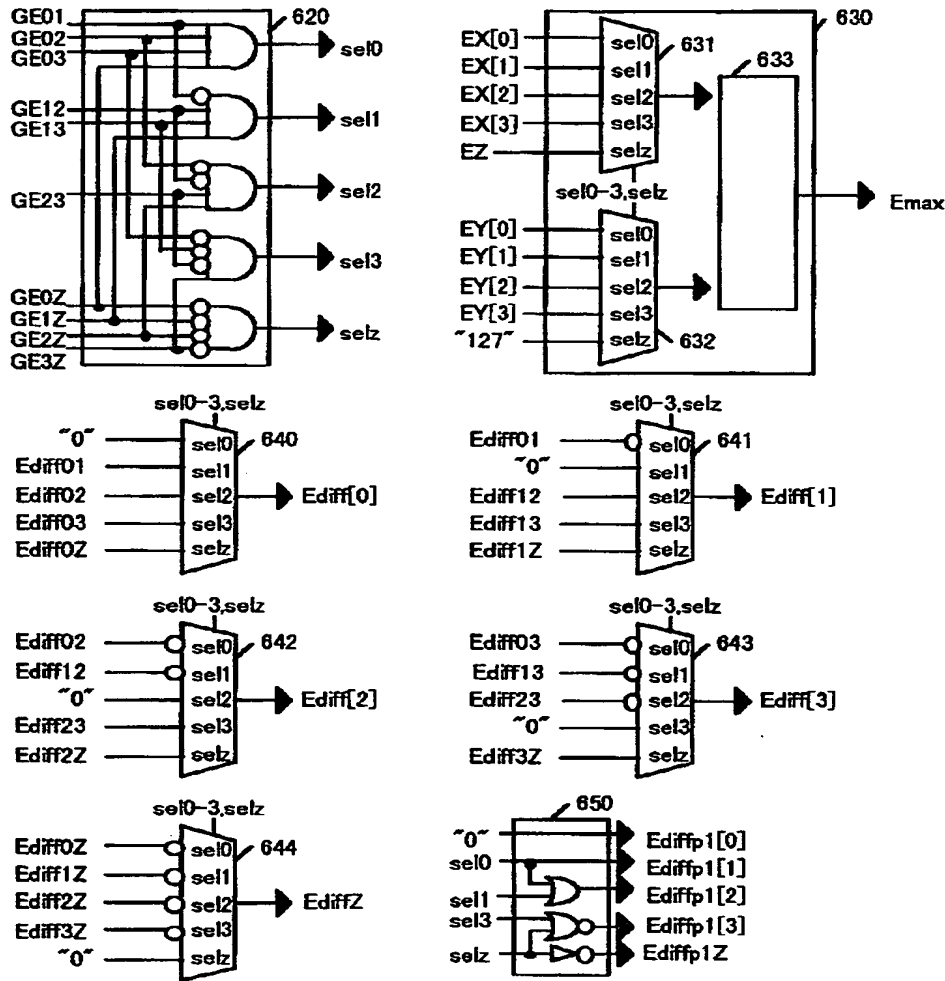
【図 8】

図 8



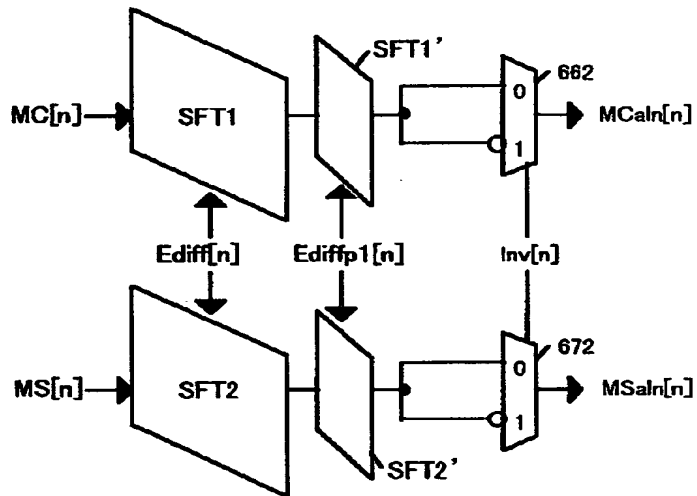
【図 9】

図 9



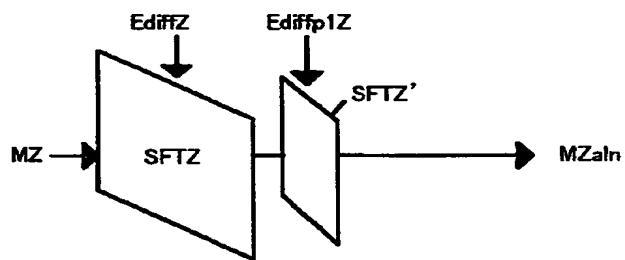
【図 1 0】

図 10



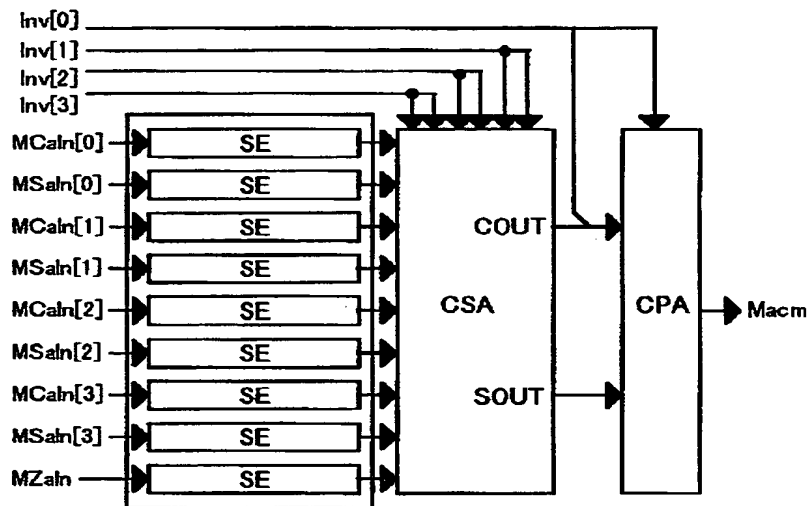
【図 1 1】

図 11



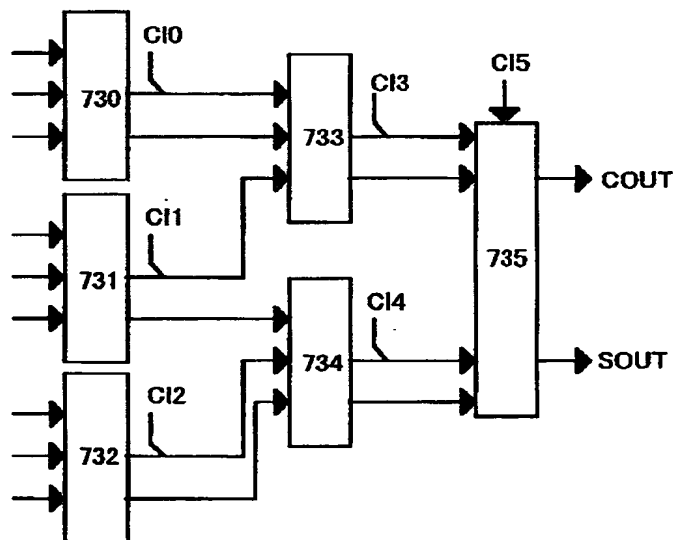
【図 1 2】

図 12



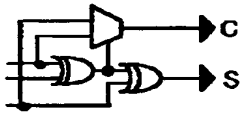
【図 1 3】

図 13



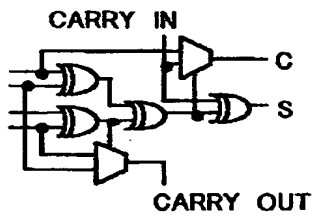
【図 1 4】

図 14



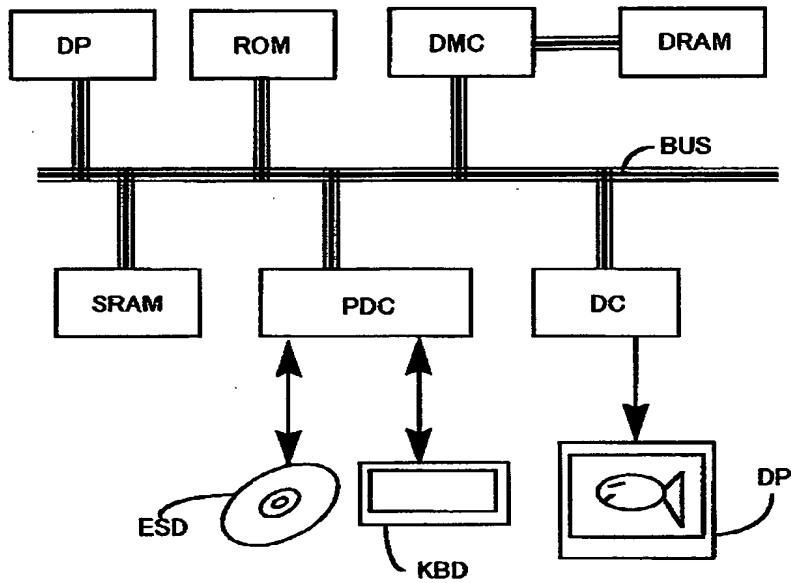
【図 1 5】

図 15



【図 1 6】

図 16



【書類名】 要約書

【要約】 浮動小数点内積演算器を SIMD 化することにより演算並列度を向上させ、演算処理能力高めたデータプロセッサの提供を可能とする。

【課題】 浮動小数点 4 元ベクトル内積演算器の効率性を維持したまま 1 命令当りの演算並列度を飛躍的に高める演算方式を実現すること。

【解決手段】 浮動小数点 4 元ベクトル内積演算器を幅広い演算系の利用が可能になっても、最小幅（単精度なら 32 ビット）で定義し、内積演算器を SIMD 化する。

【効果】 内積演算器と SIMD 化との双方による並列度向上の相乗効果で演算並列度を飛躍的に高めた。4 元ベクトルの内積とスカラとの和を求める浮動小数点演算器を 4 並列 SIMD 化すると 1 サイクル当り 32 FLOPS となる。

【選択図】 図 2

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都千代田区神田駿河台4丁目6番地
氏 名 株式会社日立製作所